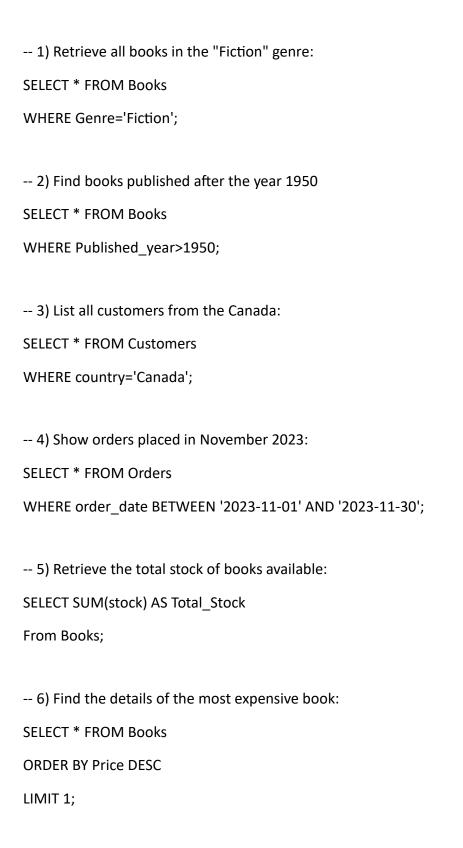
```
-- Create Database
CREATE DATABASE OnlineBookstore;
-- Switch to the database
\c OnlineBookstore;
-- Create Tables
DROP TABLE IF EXISTS Books;
CREATE TABLE Books (
  Book_ID SERIAL PRIMARY KEY,
  Title VARCHAR(100),
 Author VARCHAR(100),
  Genre VARCHAR(50),
  Published_Year INT,
  Price NUMERIC(10, 2),
  Stock INT
);
DROP TABLE IF EXISTS customers;
CREATE TABLE Customers (
  Customer_ID SERIAL PRIMARY KEY,
  Name VARCHAR(100),
 Email VARCHAR(100),
  Phone VARCHAR(15),
  City VARCHAR(50),
  Country VARCHAR(150)
);
```

```
DROP TABLE IF EXISTS orders;
CREATE TABLE Orders (
  Order_ID SERIAL PRIMARY KEY,
  Customer ID INT REFERENCES Customers (Customer ID),
  Book ID INT REFERENCES Books(Book ID),
  Order_Date DATE,
  Quantity INT,
  Total Amount NUMERIC(10, 2)
);
SELECT * FROM Books;
SELECT * FROM Customers;
SELECT * FROM Orders;
-- Import Data into Books Table
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'C:\SQL Projects\Bookstore by Satish Dhawale\Books.csv'
CSV HEADER;
-- Import Data into Customers Table
COPY Customers (Customer_ID, Name, Email, Phone, City, Country)
FROM 'C:\SQL Projects\Bookstore by Satish Dhawale\Customers.csv'
CSV HEADER;
-- Import Data into Orders Table
COPY Orders(Order ID, Customer ID, Book ID, Order Date, Quantity, Total Amount)
FROM 'C:\SQL Projects\Bookstore by Satish Dhawale\Orders.csv'
CSV HEADER;
```

## -- BASIC QUESTIONS:



-- 7) Show all customers who ordered more than 1 quantity of a book: **SELECT \* FROM Orders** WHERE quantity>1; -- 8) Retrieve all orders where the total amount exceeds \$20: SELECT \* FROM Orders WHERE total amount>20; -- 9) List all genres available in the Books table: SELECT DISTINCT genre FROM Books; -- 10) Find the book with the lowest stock: **SELECT \* FROM Books** ORDER BY stock LIMIT 1; -- 11) Calculate the total revenue generated from all orders: SELECT SUM(total\_amount) As Revenue FROM Orders; --ADVANCE QUERIES --1) Retrieve the total number of books sold for each genre SELECT b.Genre AS Genre, SUM(o.Quantity) AS Total Books Sold FROM Orders o JOIN Books b ON o.book\_id = b.book\_id

GROUP BY b.Genre;

```
--2) Find the average price of books in the "Fantasy" genre
SELECT ROUND(AVG(price),2) AS Average_Price
FROM Books
WHERE Genre='Fantasy';
--3) List customers who have placed at least 2 orders
SELECT o.customer id, c.name, COUNT(o.Order id) AS ORDER COUNT
FROM orders o
JOIN customers c ON o.customer id = c.customer id
GROUP BY o.customer_id, c.name
HAVING COUNT(o.order id)>=2;
--4) Find the most frequently ordered book
SELECT o.Book id, b.title, COUNT(o.order id) AS ORDER COUNT
FROM orders o
JOIN Books b ON b.book id = o.book id
GROUP BY o.book id, b.title
ORDER BY ORDER_COUNT DESC LIMIT 1;
--5) Show the top 3 most expensive books of 'Fantasy' Genre
SELECT * FROM books
WHERE genre='Fantasy'
ORDER BY price DESC LIMIT 3;
--6) Retrieve the total quantity of books sold by each author
SELECT b.author, SUM(o.quantity) AS TOTAL_BOOKS_SOLD
FROM Orders o
JOIN Books b ON o.book id = b.book id
GROUP BY b.author;
```

--7) List the cities where customers who spent over \$30 are located SELECT DISTINCT c.City, o.total\_amount FROM orders o JOIN customers c ON c.customer id=o.customer id WHERE Total Amount>30; --8) Find the customer who spent the most on orders SELECT DISTINCT c.Customer id, c.Name, SUM(o.total amount) AS TOTAL SPENT FROM orders o JOIN customers c ON c.customer\_id=o.customer\_id GROUP BY c.Customer id, c.Name ORDER BY TOTAL\_SPENT DESC LIMIT 1; --9) Calculate the stock remaining after fulfilling all order SELECT b.book id, b.title,b.stock, COALESCE(SUM(o.quantity),0) AS Order Quantity, b.stock - COALESCE(SUM(o.Quantity),0) AS Remaining\_Quantity from Books b LEFT JOIN Orders o ON b.book id=o.book id

GROUP BY b.book\_id

ORDER BY b.book id;